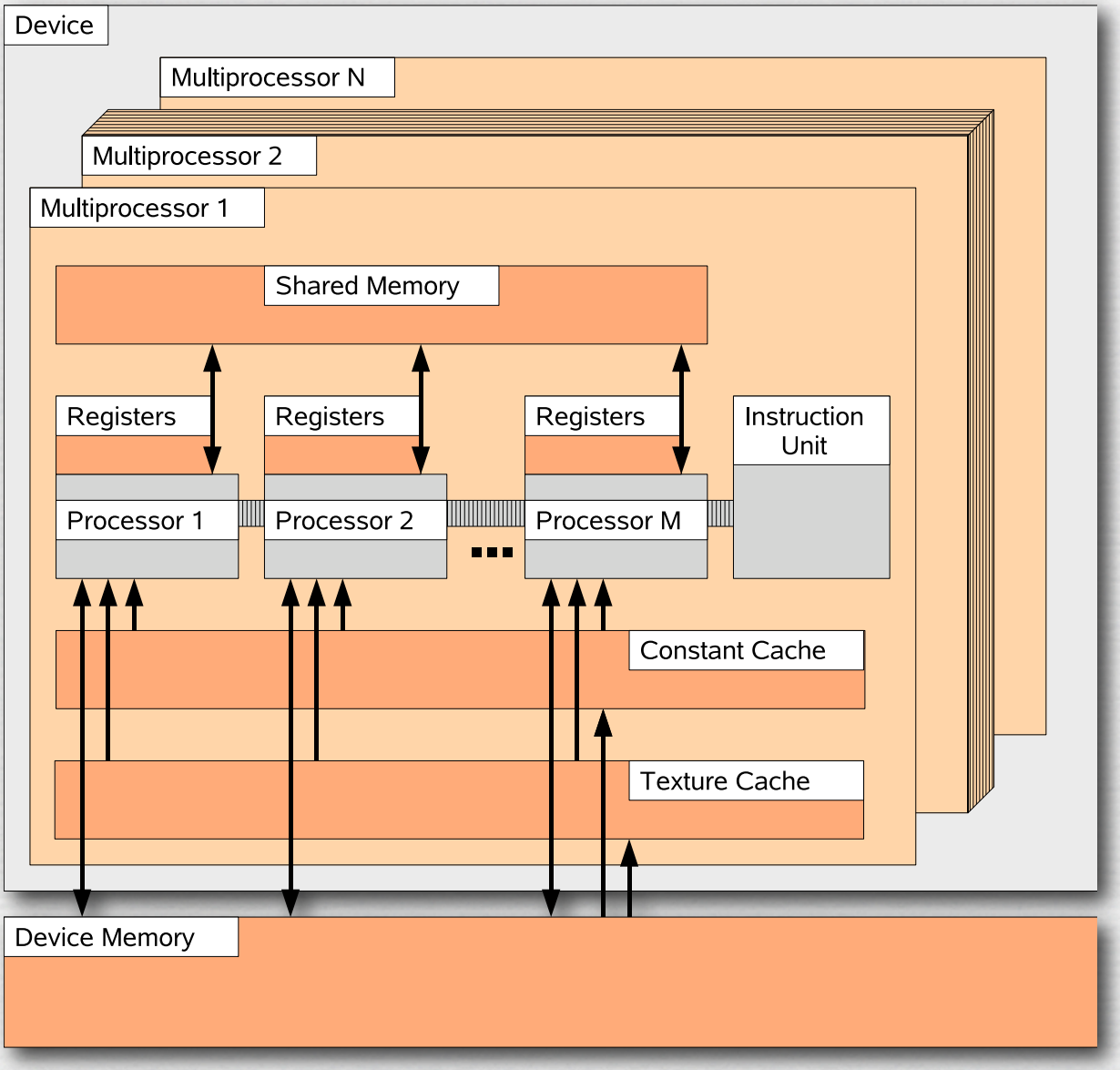
Accelerated Fluctuation Analysis by Graphic Cards and Complex Pattern Formation in Econophysics

Tobias Preis

Institute of Physics, Johannes Gutenberg-University of Mainz
Center for Polymer Studies, Department of Physics, Boston University
Artemis Capital Asset Management GmbH
preis@uni-mainz.de

With thanks to: Peter Virnau, Wolfgang Paul, and Johannes J. Schneider

Reminder: GPU device architecture



GeForce GTX 280:

Global memory 1024 MB

Number of multiprocessors 30

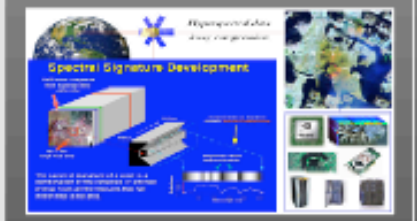
Number of cores 240

Constant memory 64 kB

Shared memory 16 kB

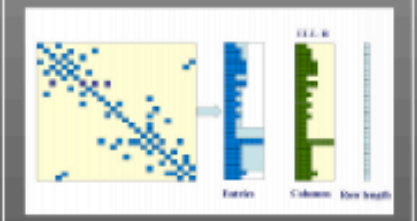
Clock rate 1.30 GHz

LATEST CUDA NEWS CSI-Style Video Enhancement For Consumers



Hyperspectral image compression on NVidia GPUs

26 x



The sparse matrix vector product on GPUs

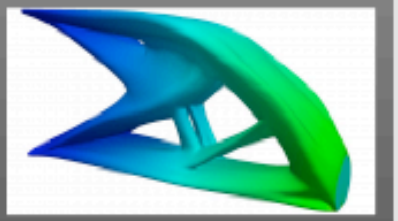
80 x

cRARk

15 x

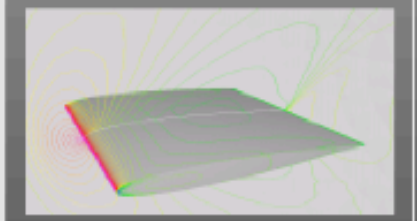
Density field viewer

10 x



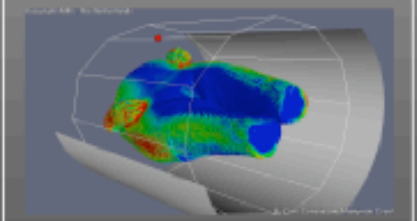
GPUTop - Topology Optimization on CUDA Graphics Cards in 3D

60 x



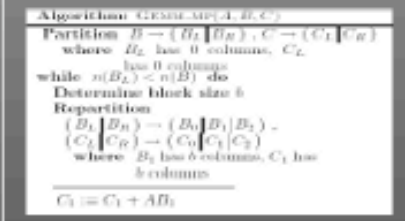
Running Unstructured Grid CFD Solvers on Modern Graphics Hardware

33 x



Parallel Computation With NVIDIA Graphics Card Using CUDA in Hyperthermia

25 x

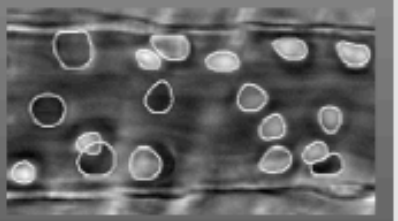


Level-3 BLAS on a GPU: Picking the Low Hanging Fruit

4 x

Parallelized Turing bombe & Enigma simulations

35 x

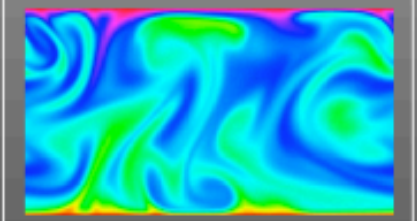


Accelerating Leukocyte Tracking using CUDA

29 x

Real Time Holographic Optical Trapping

350 x

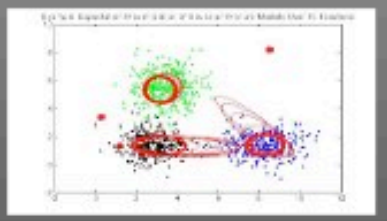


A Fast Double Precision CFD Code using CUDA

8 x

Clustering Billions of Data Points Using GPUs

300 x



CUDA Accelerated Expectation Maximization of Gaussian Mixture Models

170 x

GPU accelerated Monte Carlo simulation of the Ising model

60 x

DOWNLOAD AND INSTALLATION TIPS

NVIDIA's CUDA development tools provide three key components to help you get started: the latest CUDA driver, a complete CUDA Toolkit, and CUDA SDK code samples.

Please download files and install them in this order:

Source: http://www.nvidia.com/cuda

1. CUDA Driver **2. CUDA Toolkit** **3. CUDA SDK code samples**

The CUDA driver and Toolkit installation are required before running the precompiled examples or compiling the example source code.

* An updated version of the CUDA Toolkit end user license agreement (EULA) that specifically allows redistribution of the runtime and libraries included in the toolkit is now available.

CHOOSE YOUR OS

ed only on 32-bit and 64-bit Red Hat Enterprise Linux

Apply now!

How to start ...?

eate applications for a wide range of NVIDIA GPUs –
o Quadro® and Tesla™ GPUs for professional
UDA-enabled products.

How to start ...?

CUDA 2.2 FOR LINUX

Source: http://www.nvidia.com/cuda

CUDA 2.2 QuickStart Guide	Download
CUDA 2.2 Programming Guide	Download
CUDA 2.2 Reference Manual	Download
CUDA 2.2 Toolkit Release Notes	Download
CUDA 2.2.1 SDK Release Notes	Download
CUDA Visual Profiler v1.2 Readme	Download
Documentation for CUDA BLAS (CUBLAS) Library	Download
Documentation for CUDA FFT (CUFFT) Library	Download

CUDA Plug-in Documentation

MathWorks MATLAB® Plug-in	Download
CUDA Photoshop Plug-ins (documentation): Source code examples for Windows and Mac OS (for CUDA 1.1 and later)	Download
CUDA Photoshop Plug-ins: Documentation (for CUDA 1.1 and later)	Download

How to compile ...?

Path Variables:

export LD_LIBRARY_PATH=/usr/local/cuda/lib:
export PATH=/usr/local/cuda/bin:$PATH

NVCC Compiler:

nvcc -O3 main.cu -I/home/tobias/NVIDIA_CUDA_SDK/common/inc
-L/home/tobias/NVIDIA_CUDA_SDK/lib -lcutil

Double Precision:
-arch sm_13

CUDA Example

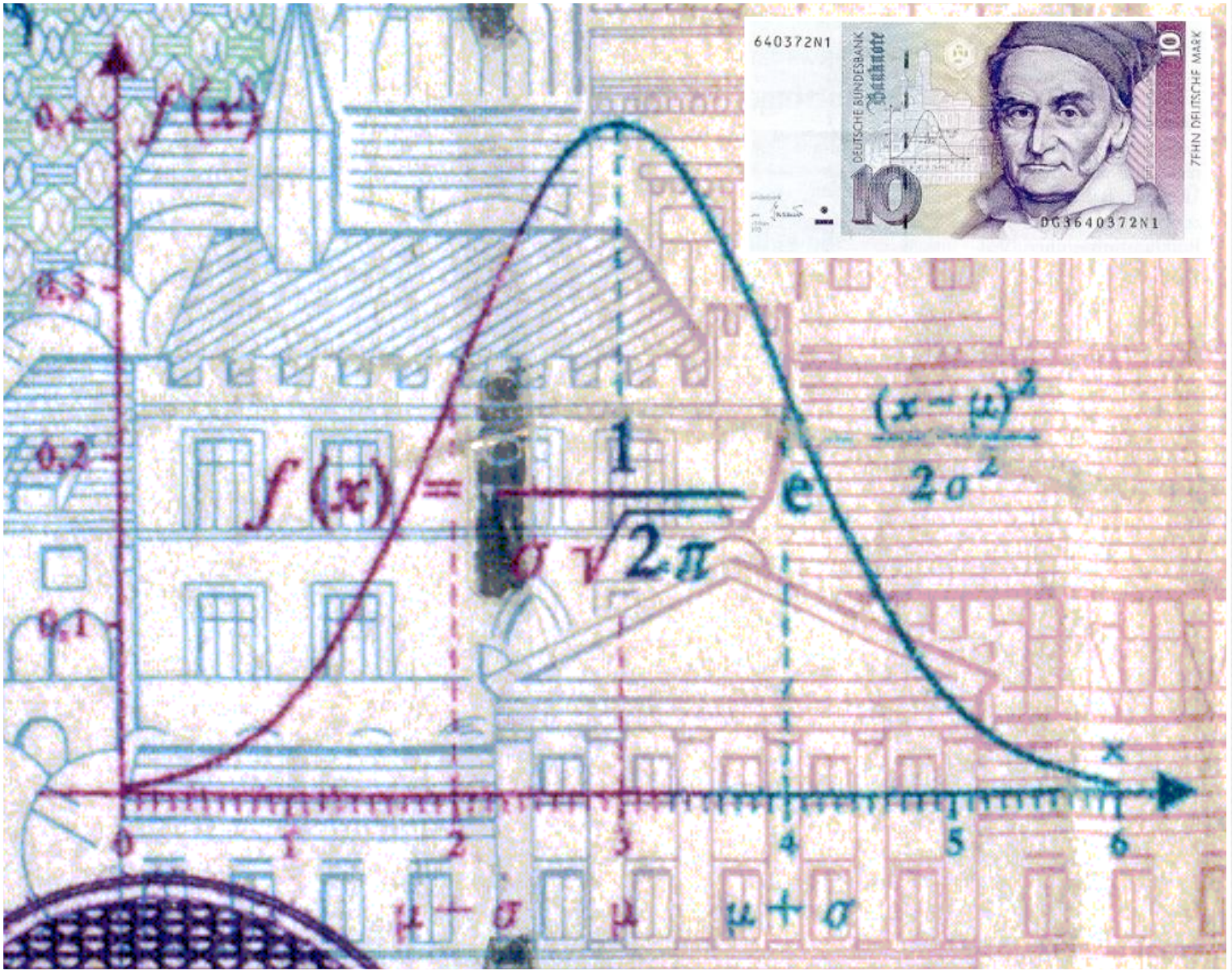
Array manipulation

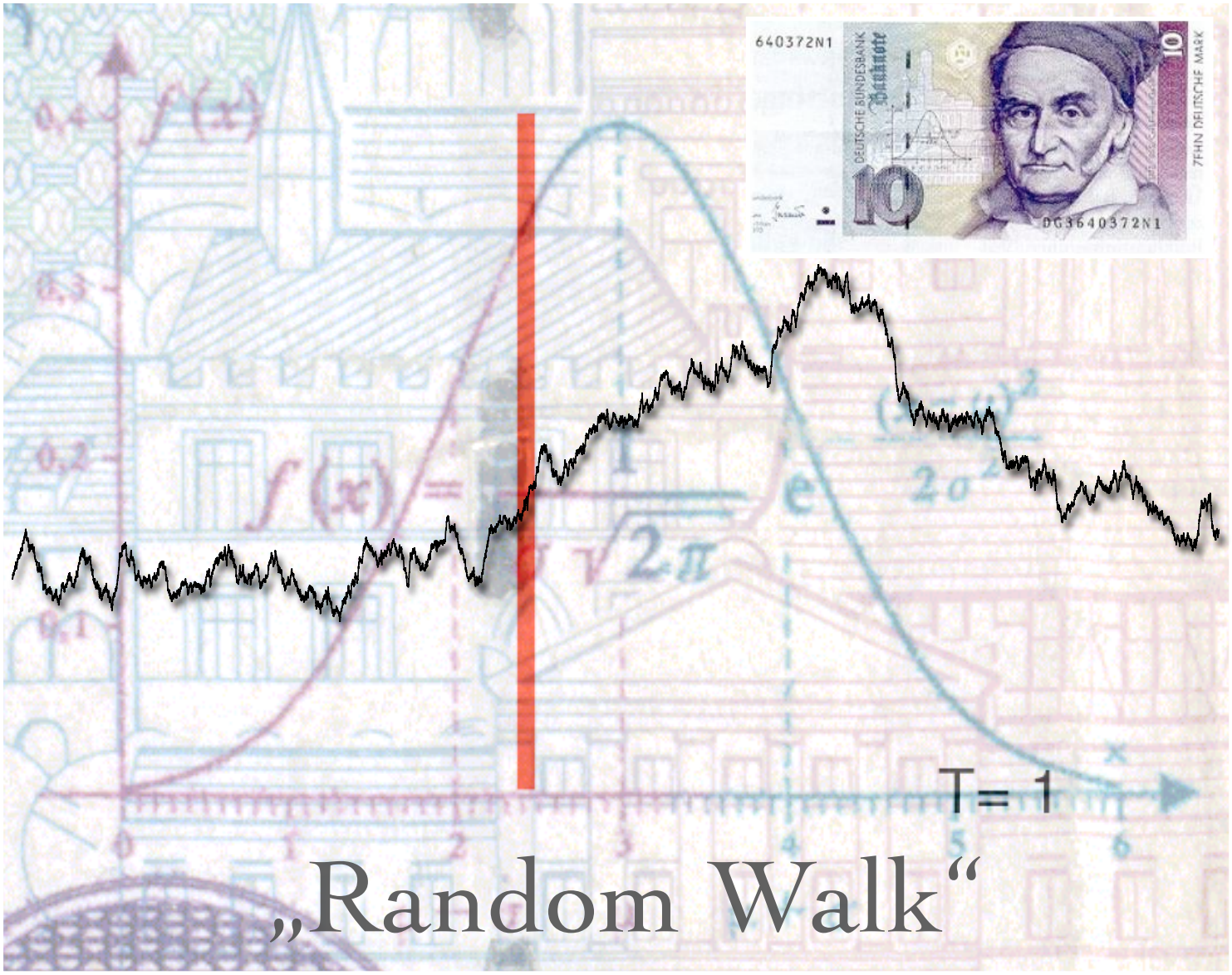
```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <cutil.h>

#define BLOCK_SIZE 32

__global__ void gpu(float* x) {
  int id=threadIdx.x+BLOCK_SIZE*blockIdx.x;
  x[id]+=1;
}

int main(int argc,char** argv) {

  CUT_DEVICE_INIT(argc,argv);
  unsigned int mem_size=sizeof(float)*512;
  float* x=(float*) malloc(mem_size);
  for(int i=0; i<512;i++) x[i]=1;

  float* d_x;
  CUDA_SAFE_CALL(cudaMalloc((void**) &d_x,mem_size));
  CUDA_SAFE_CALL(cudaMemcpy(d_x,x,mem_size,cudaMemcpyHostToDevice));

  //GPU Function
  gpu<<<16,BLOCK_SIZE>>>(d_x);
  CUDA_SAFE_CALL(cudaMemcpy(x,d_x,mem_size,cudaMemcpyDeviceToHost));

  float sum=0;
  for(int i=0; i<512;i++) sum+=x[i];
  printf("%f \n",sum);
  CUDA_SAFE_CALL(cudaFree(d_x));
}
```

GPGPU / Time Series Analysis

Tobias Preis

GPGPU / Time Series Analysis




640372N1
DEUTSCHE BUNDESBANK
Banknote
10
ZEHN DEUTSCHE MARK
DG3640372N1
T= 1

„Random Walk“

German Stock Index (Dax)

20.06.2008

15.09.2008

LEHMAN BROTHERS

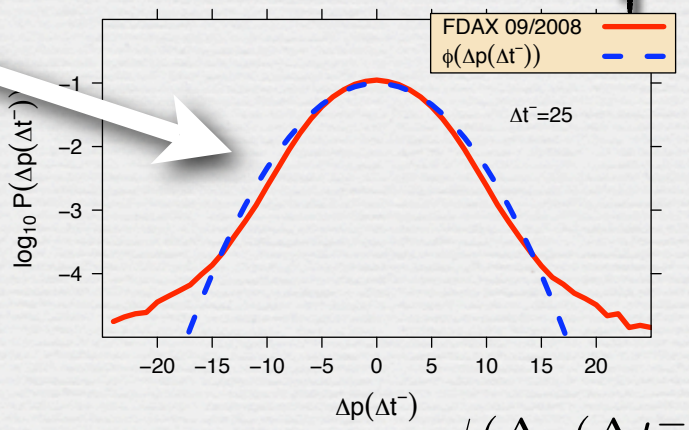
19.09.2008

PDF of returns

Hurst Exponent

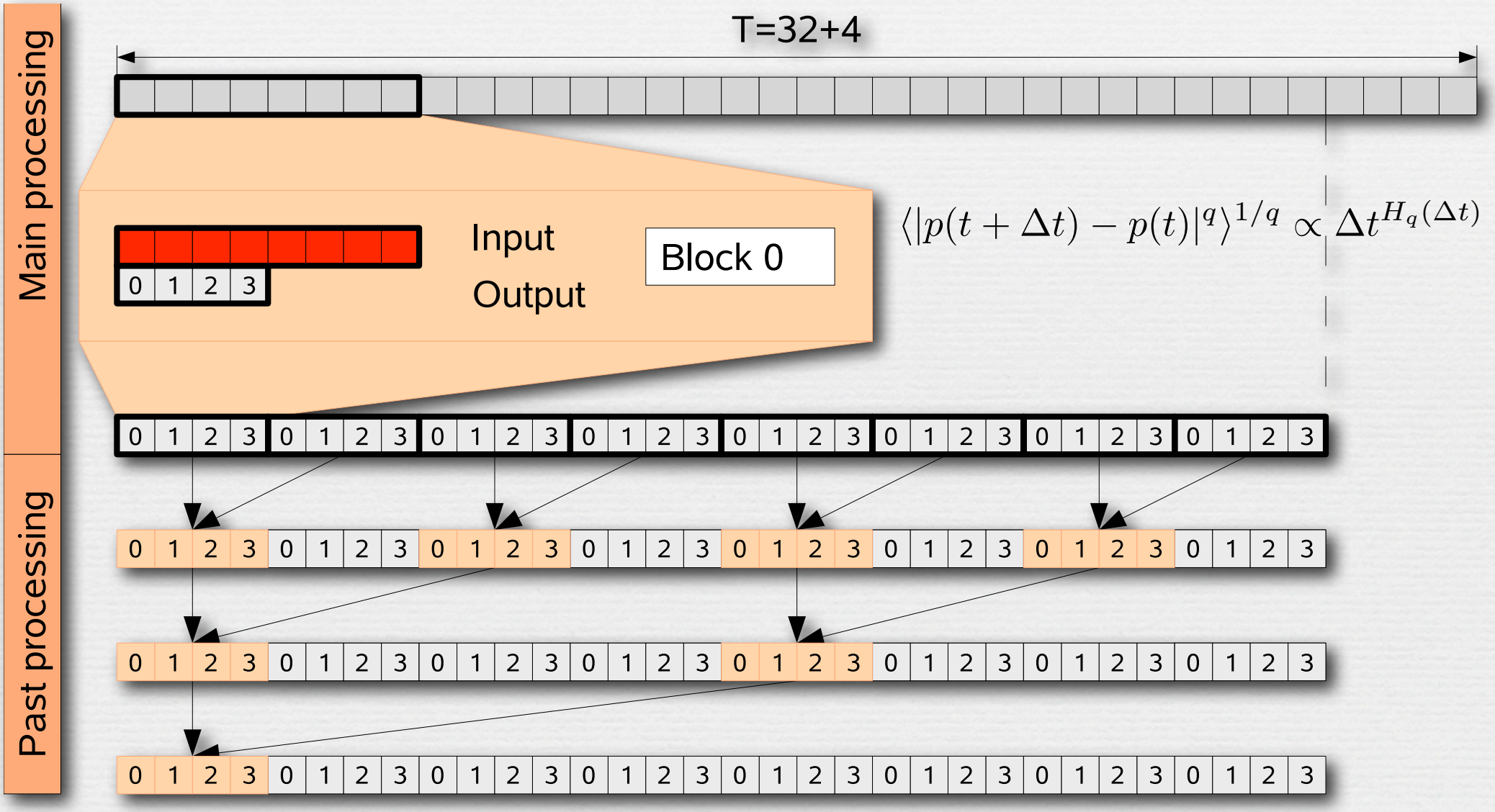
Autocorrelation

Pattern Conformity

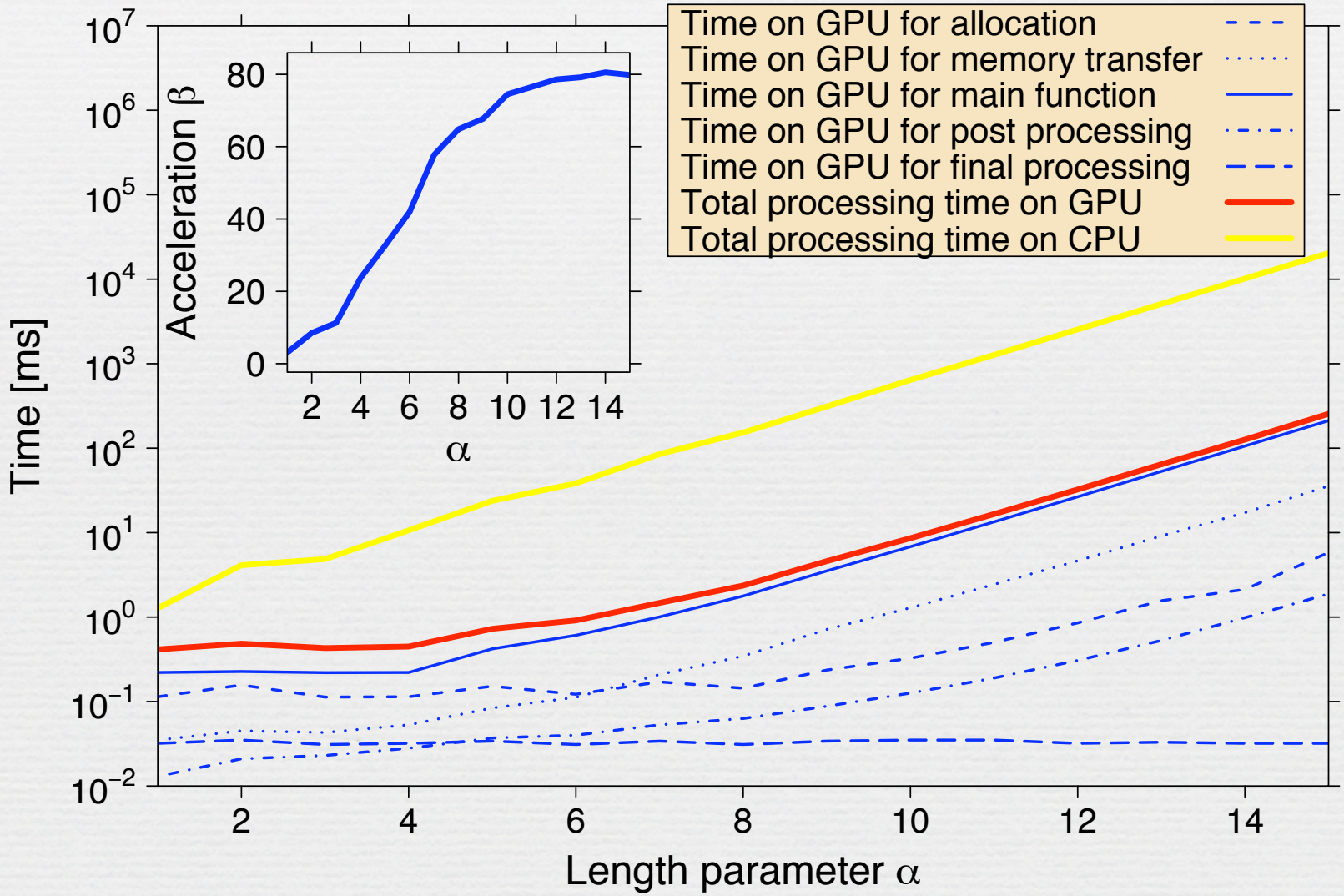


$$\phi(\Delta p(\Delta t^-)) = u \exp(-v \Delta p^2)$$

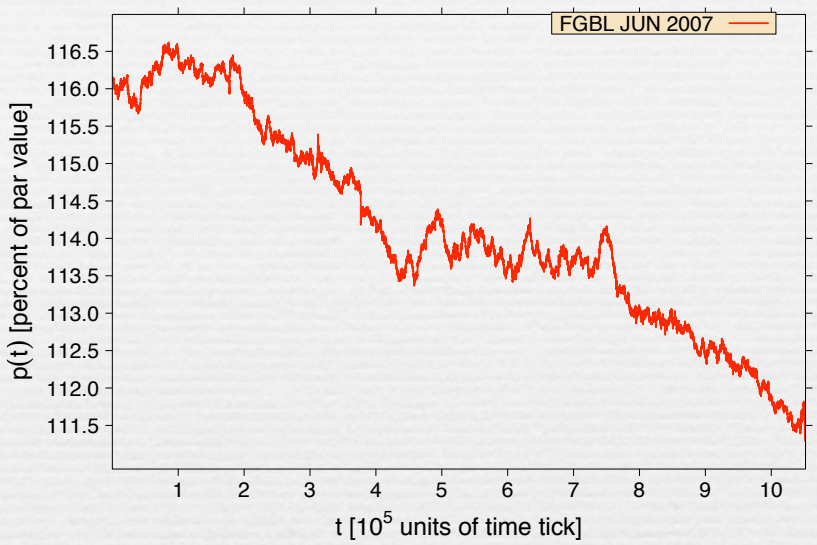
GPU computing / Hurst exponent

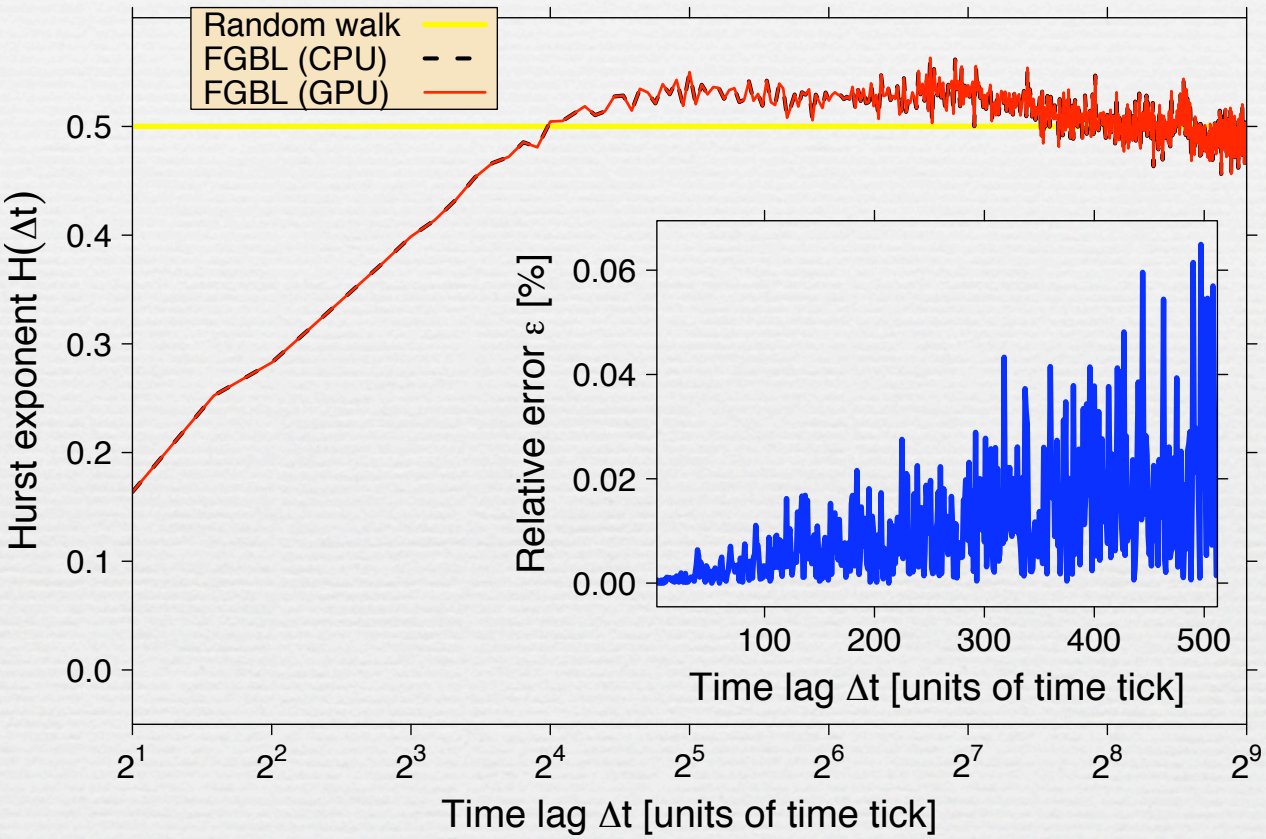


GPU computing / Hurst exponent



GPU computing / Hurst exponent

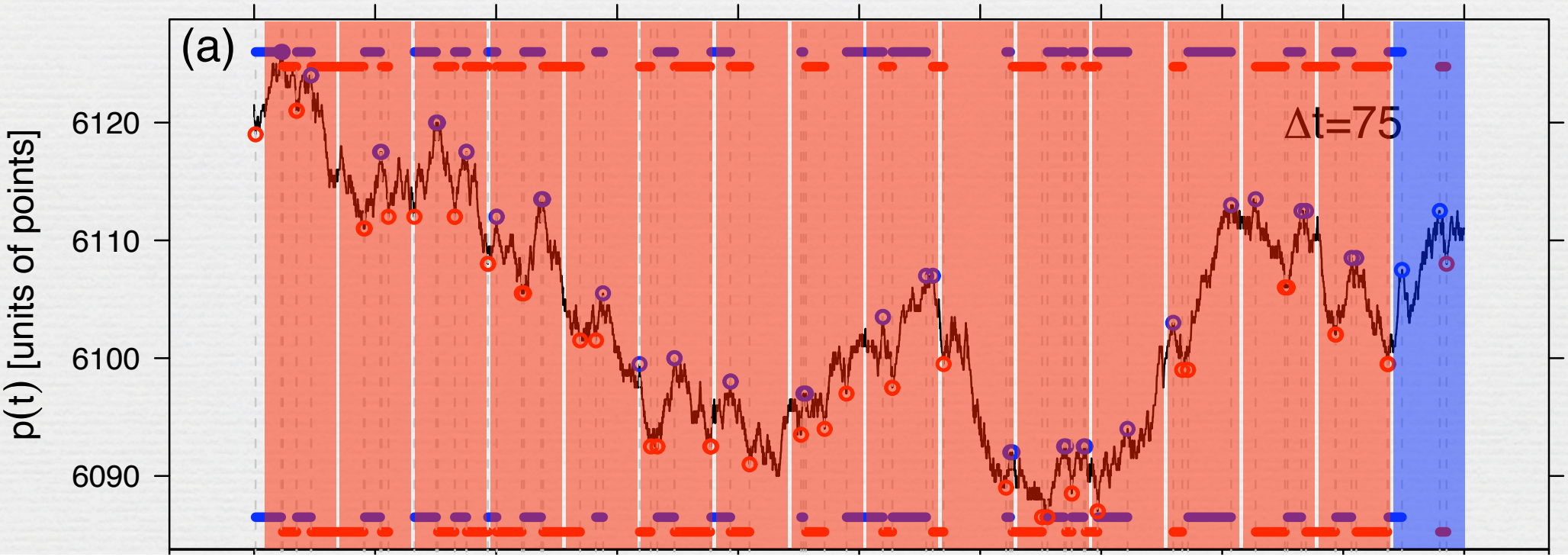




TP, P. Virnau, W. Paul, and J. J. Schneider, Preprint submitted (2009)

Fluctuation Patterns

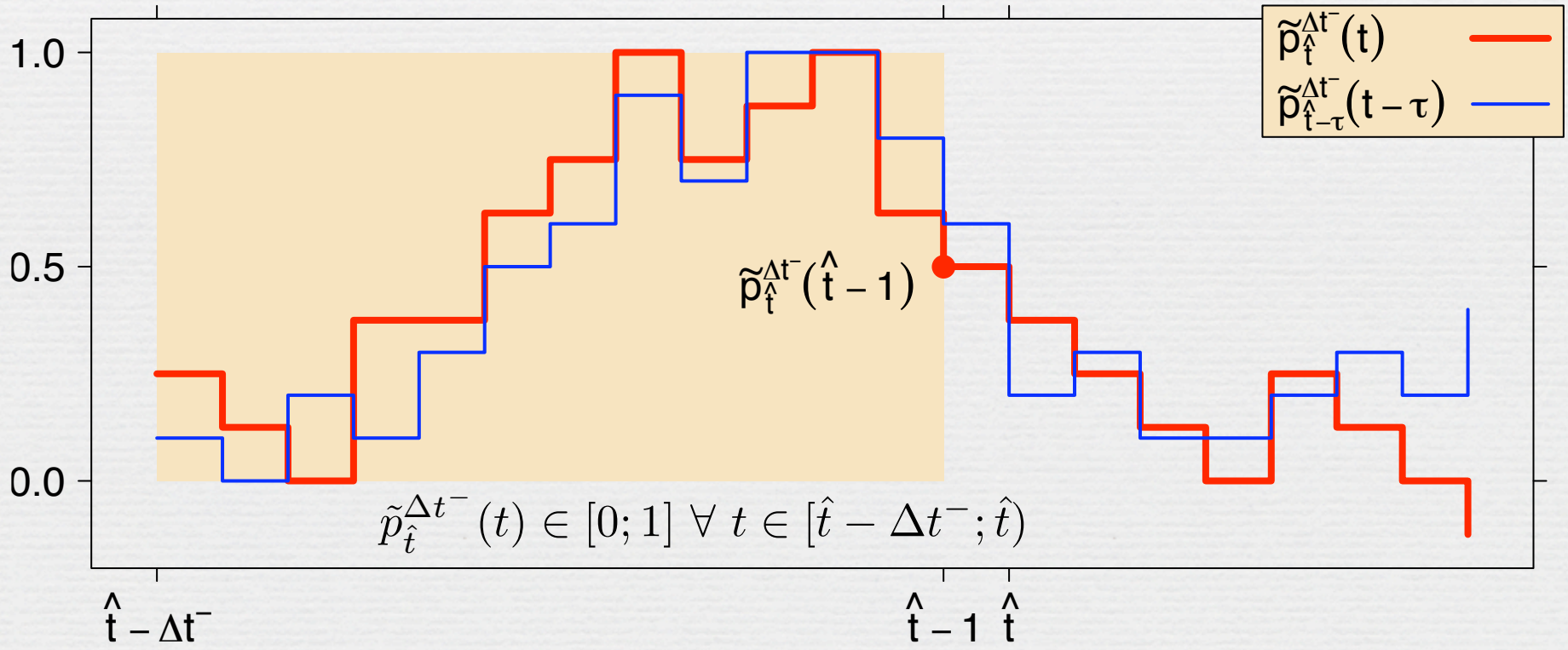
„The aim is to compare the current reference pattern of time interval length Δt^- with all previous patterns in the time series.“



Fluctuation Patterns

True range adapted modified time series $\tilde{p}_{\hat{t}}^{\Delta t^-}(t)$

$$\tilde{p}_{\hat{t}}^{\Delta t^-}(t) = \frac{p(t) - p_l(\hat{t}, \Delta t^-)}{p_h(\hat{t}, \Delta t^-) - p_l(\hat{t}, \Delta t^-)}$$

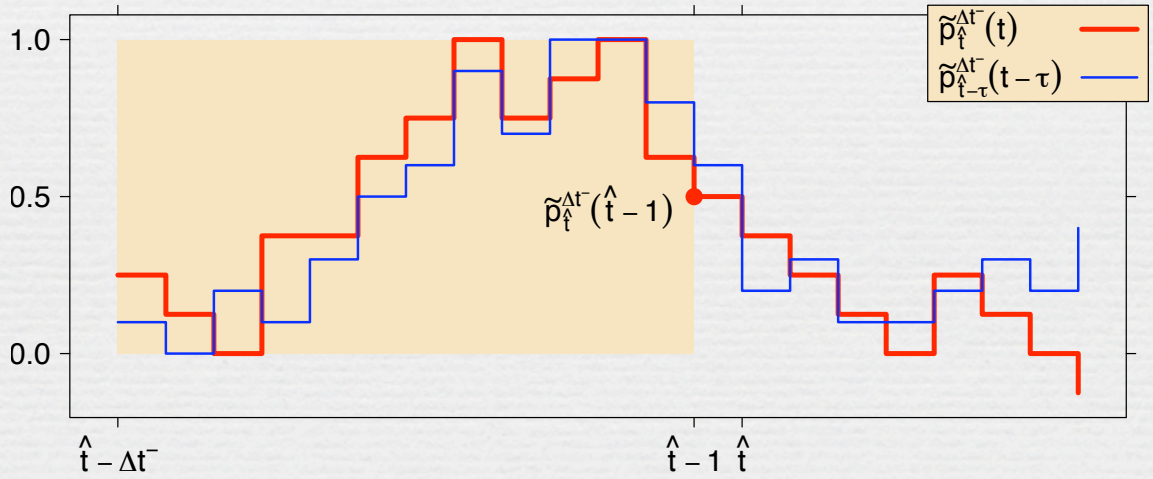


Fluctuation Patterns

Mean-square quality between current and comparison sequence

$$Q_{\hat{t}}^{\Delta t^-}(\tau) = \sum_{\theta=1}^{\Delta t^-} \frac{\left(\tilde{p}_{\hat{t}}^{\Delta t^-}(\hat{t}-\theta) - \tilde{p}_{\hat{t}-\tau}^{\Delta t^-}(\hat{t}-\tau-\theta)\right)^2}{\Delta t^-}$$

with $Q_{\hat{t}}^{\Delta t^-}(\tau) \in [0,1]$



In order to quantify the value of reference and comparison pattern relative to the reference point, one can define ...

$$\omega_{\hat{t}}^{\Delta t^-}(\tau,\Delta t^+) = \left(\tilde{p}_{\hat{t}}^{\Delta t^-}(\hat{t}-1+\Delta t^+) - \tilde{p}_{\hat{t}}^{\Delta t^-}(\hat{t}-1)\right)\cdot\left(\tilde{p}_{\hat{t}-\tau}^{\Delta t^-}(\hat{t}-\tau-1+\Delta t^+) - \tilde{p}_{\hat{t}}^{\Delta t^-}(\hat{t}-1)\right)$$

Fluctuation Patterns

Observable for pattern conformity:

$$\xi_\chi\left(\Delta t^+,\Delta t^-\right) = \sum_{\hat{t}=\Delta t^-}^{T-\Delta t^+} \sum_{\tau=\tau^*}^{\hat{t}} \frac{\operatorname{sgn}\left(\omega_{\hat{t}}^{\Delta t^-}\left(\tau,\Delta t^+\right)\right)}{\exp\left(\chi Q_{\hat{t}}^{\Delta t^-}(\tau)\right)}$$

Limitation: $\tau^* = \begin{cases} \hat{t}-\hat{\tau} & \text{if } \hat{t}-\hat{\tau}-\Delta t^- \geq 0 \\ \Delta t^- & \text{else} \end{cases}$

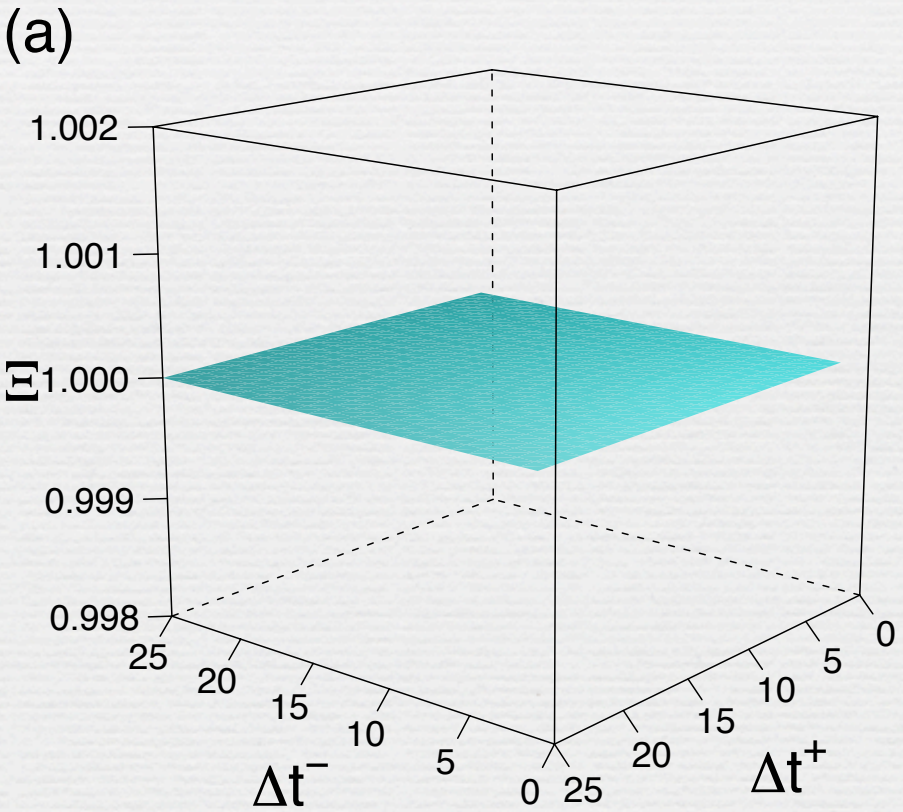
Definition: $\operatorname{sgn}(x) = \begin{cases} 1 & \text{for } x > 0 \\ 0 & \text{for } x = 0 \\ -1 & \text{for } x < 0 \end{cases}$

Normalized pattern conformity:

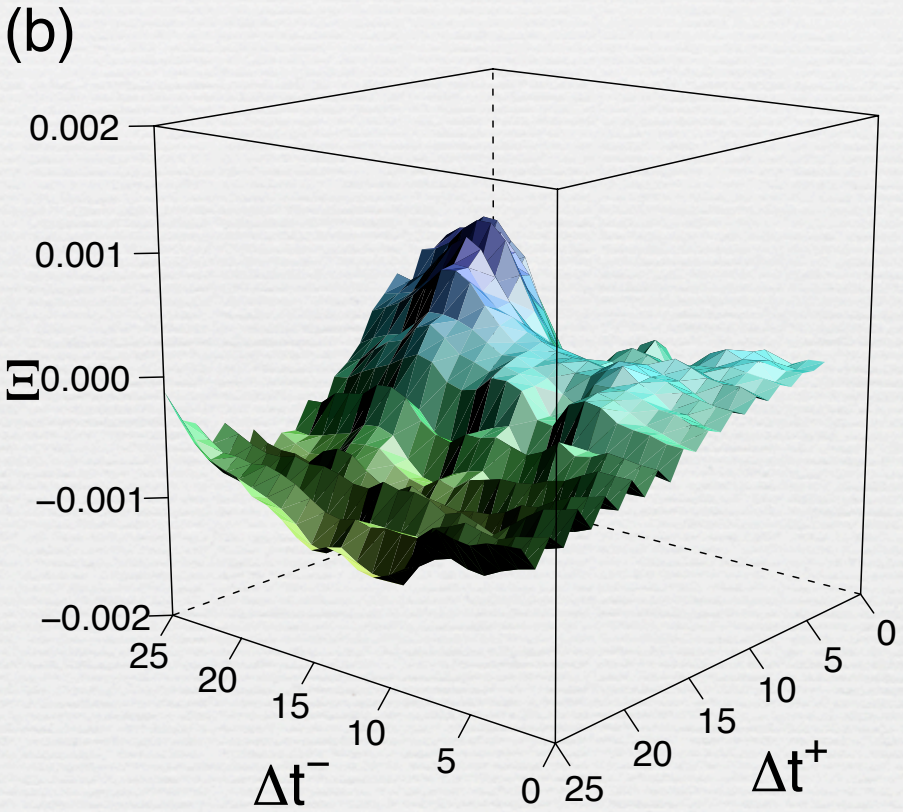
$$\Xi_\chi\left(\Delta t^+,\Delta t^-\right) = \frac{\xi_\chi\left(\Delta t^+,\Delta t^-\right)}{\sum_{\hat{t}=\Delta t^-}^{T-\Delta t^+} \sum_{\tau=\tau^*}^{\hat{t}} \frac{\left|\operatorname{sgn}\left(\omega_{\hat{t}}^{\Delta t^-}\left(\tau,\Delta t^+\right)\right)\right|}{\exp\left(\chi Q_{\hat{t}}^{\Delta t^-}(\tau)\right)}}$$

Pattern Conformity / Trivial Cases

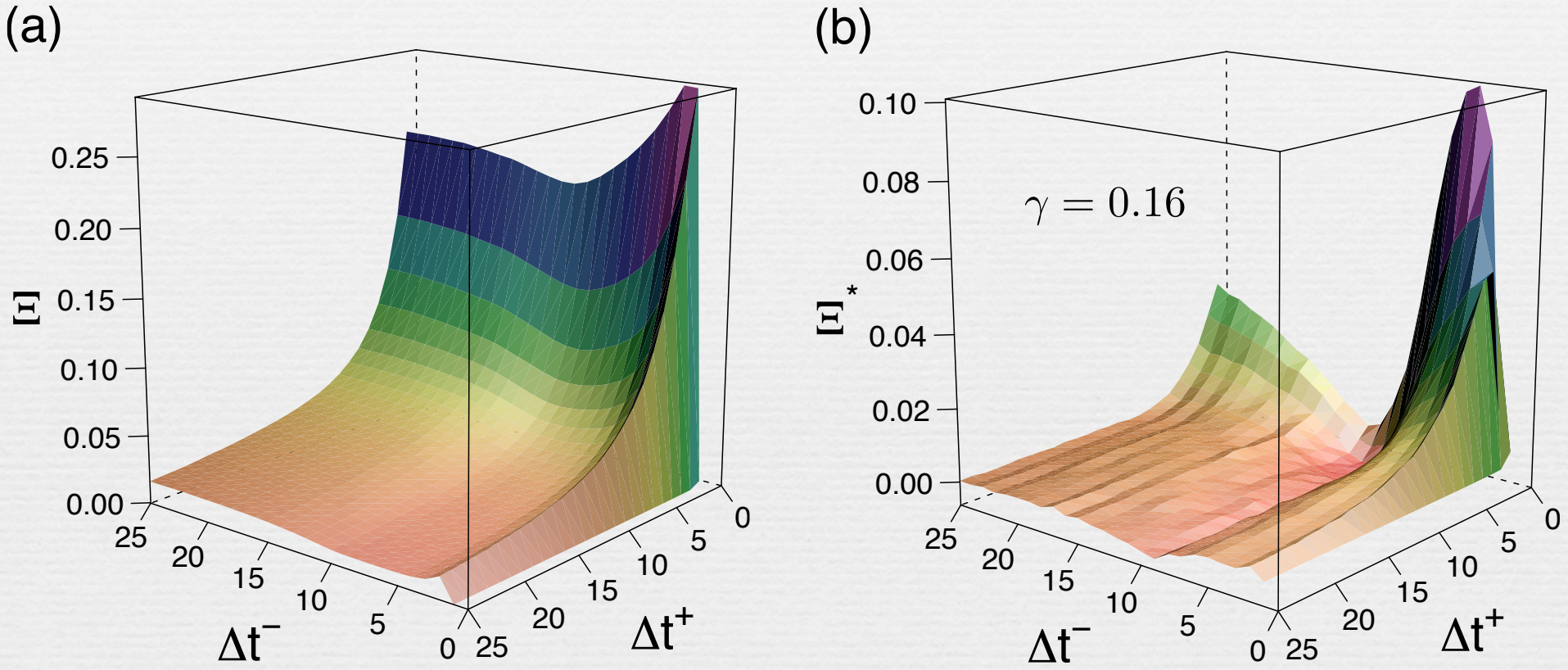
- Straight Line



- Random Walk



Pattern Conformity / FDAX



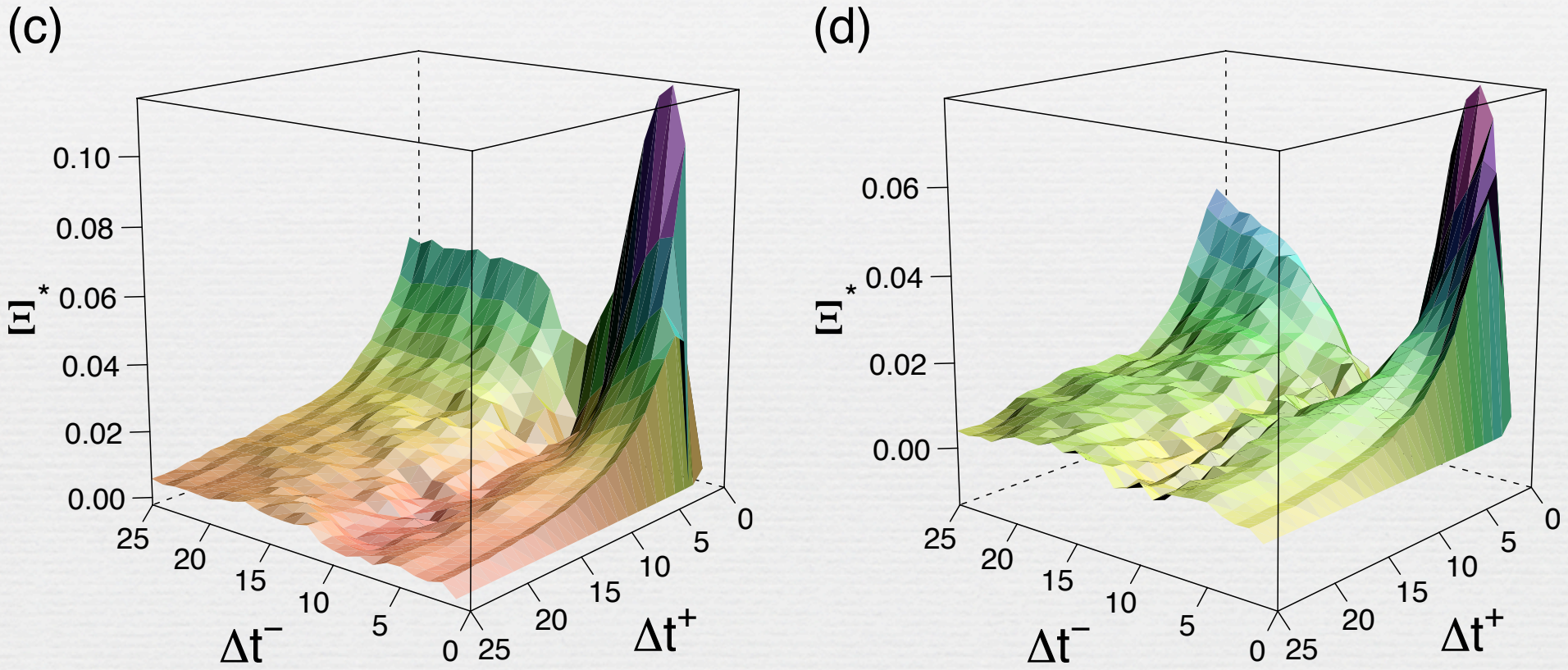
Complex correlations for financial market time series – especially for large pattern lengths.

$$\Xi^{*} = \Xi^{\mathrm{FDAX}}_{\chi=100} - \Xi^{\mathrm{ACRW}}_{\chi=100}$$

$$Q^{\Delta t^{-}}_{\hat{t}}(\tau) = Q^{p,\Delta t^{-}}_{\hat{t}}(\tau)$$

Inclusion of volumes and ITWT

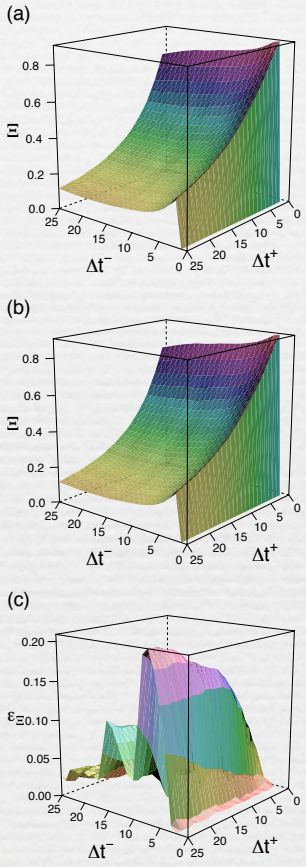
T. Preis et al., Europhys. Lett. **82**, 68005 (2008)

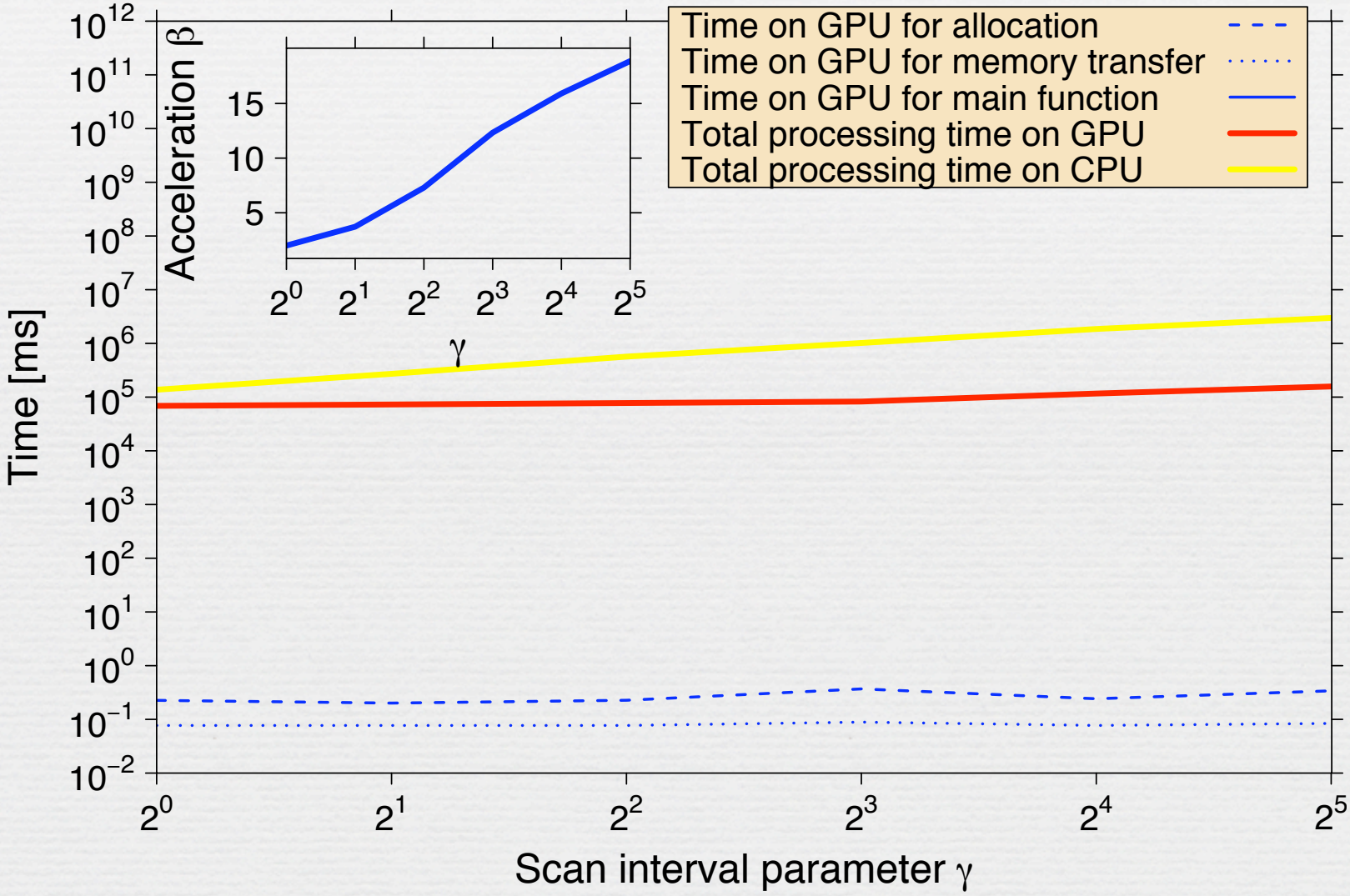


Same structure – high values of the pattern conformity

$$Q_{\hat{t}}^{\Delta t^-}(\tau) = Q_{\hat{t}}^{p,\Delta t^-}(\tau) + Q_{\hat{t}}^{v,\Delta t^-}(\tau) \qquad Q_{\hat{t}}^{\Delta t^-}(\tau) = Q_{\hat{t}}^{p,\Delta t^-}(\tau) + Q_{\hat{t}}^{\iota,\Delta t^-}(\tau)$$

GPU computing / Pattern Conformity





Final remarks

- TP, PV, WP, and JJS, „Accelerated Fluctuation Analysis by Graphic Cards and Complex Pattern Formation in Econophysics“, Preprint submitted (2009)

Thank you!